(12) UK Patent Application (19) GB (11) 2 304 428 (13) A

(43) Date of A Publication 19.03.1997

(21) Application No 9605873.0

(22) Date of Filing 20.03.1996

(30) Priority Data
(31) 07210592 (32) 18.08.1995 (33) JP

(71) Applicant(s)
Mitsubishi Denki Kabushiki Kaisha

(Incorporated in Japan)

No. 2-3 Marunouchi 2-chome, Chiyoda-ku, Tokyo 100, Japan

(72) Inventor(s)
Shinichi Mizoguchi

(74) Agent and/or Address for Service
Marks & Clerk
57-60 Lincoln's Inn Fields, LONDON, WC2A 3LS, United Kingdom

(51) INT CL[6]
G06F 11/20

(52) UK CL (Edition O)
G4A AEF A12C A12D A12T

(56) Documents Cited
EP 0675502 A2 EP 0618535 A2 EP 0617363 A2
EP 0597706 A2 EP 0590809 A2 EP 0522780 A2
EP 0392895 A2

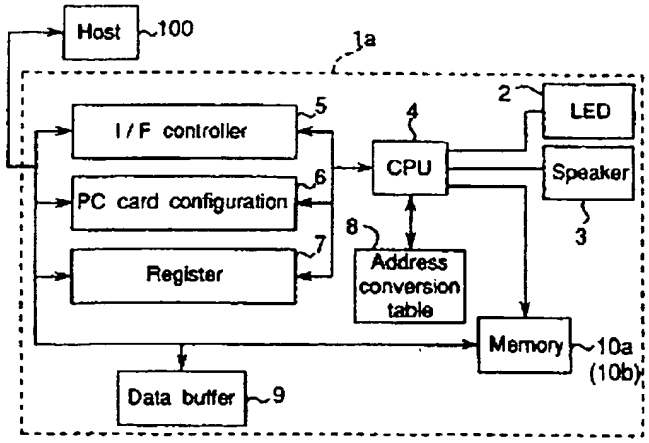
(58) Field of Search
UK CL (Edition O) G4A AEF
INT CL[6] G06F 3/06 11/20
Online: WPI

(54) Flash memory device

(57) A memory device 1a includes a flash memory 10a, 10b, comprising a plurality of blocks, each block comprising a plurality of sectors 24, the block being a unit of erasure of data. An address in a sector 24 in the flash memory 10a, 10b is identified according to address conversation in correspondence to a logical address. If the data cannot be written in a first sector 24 at the address identified, the data is written in a second sector 24 in the same block as the first sector 24, without stopping due to the write error. Then, a physical address in the flash memory 10a, 10b is rewritten in correspondence to the logical address in the address conversion table to an address of the second sector 24.

Similarly, a flash memory 20 comprises a plurality of sectors 55, if the data cannot be written in a first sector 55 at the address identified, the data is written in a second sector 55 different from the first sector 55.

Fig.2



GB 2 304 428 A

Fig. 1

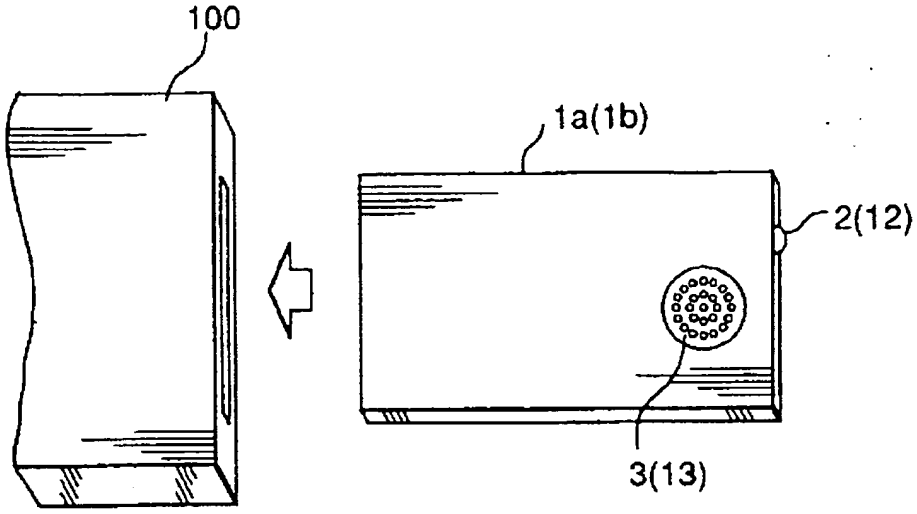


Fig.2

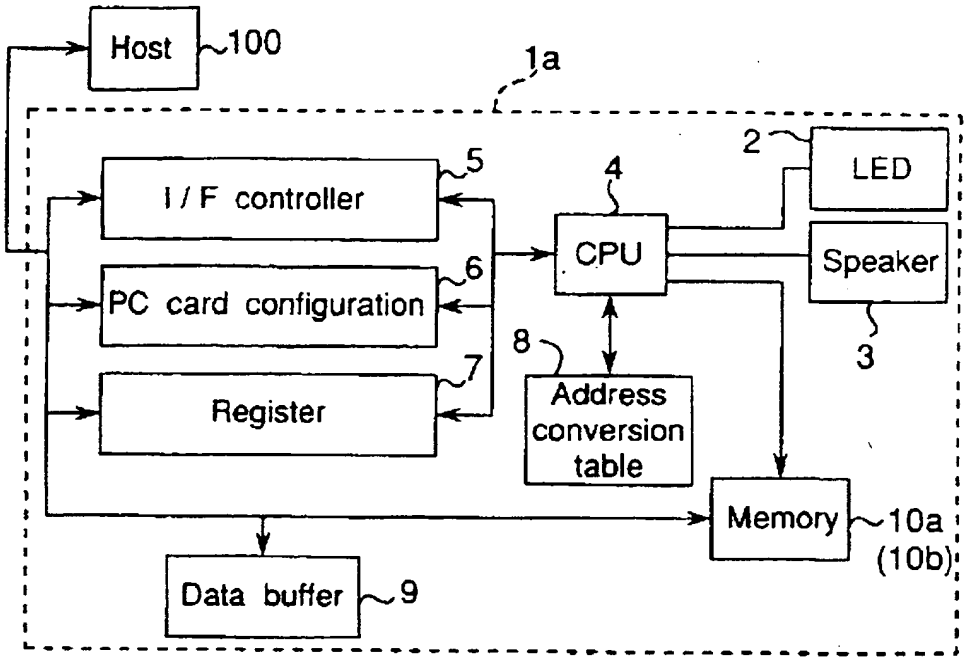

Host
100
1a
2
LED
5
I / F controller
4
CPU
Speaker
6
PC card configuration
3
8
7
Register
Address conversion table
Memory
10a
(10b)
Data buffer
9

Fig.3

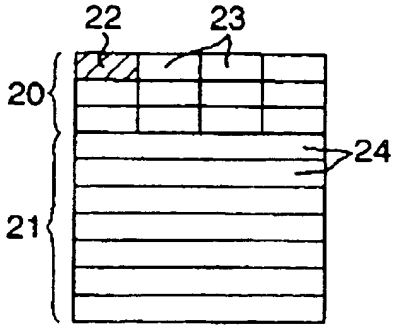

22
23
20
24
21

Fig.4

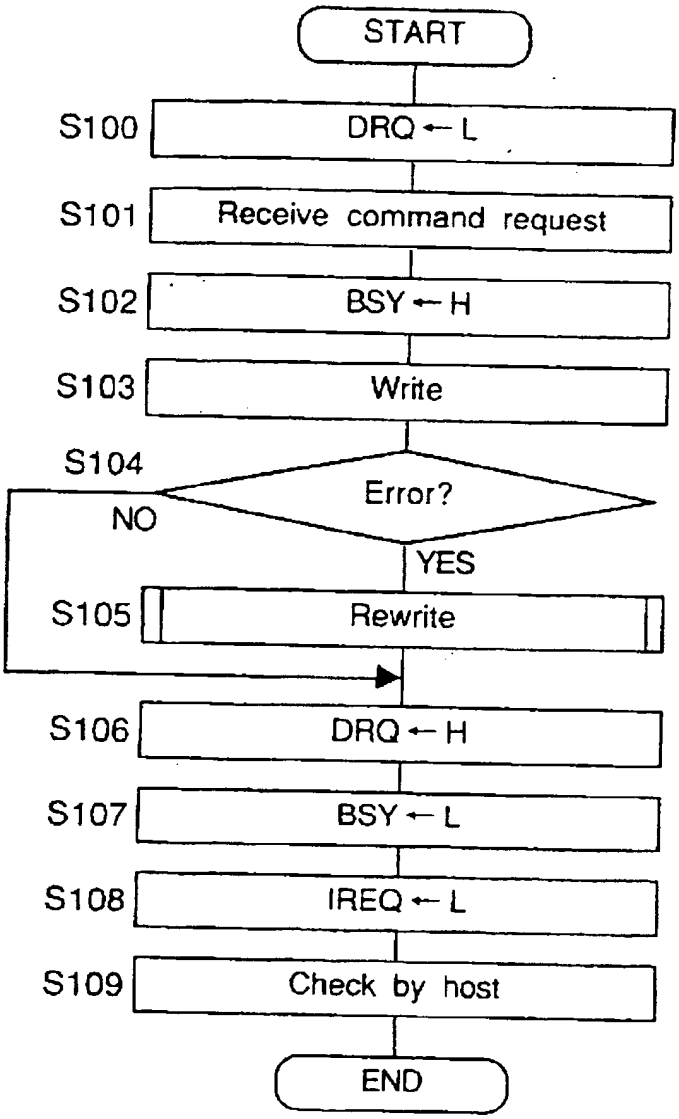

START
S100
DRQ ← L
S101
Receive command request
S102
BSY ← H
S103
Write
S104
Error?
NO
YES
S105
Rewrite
S106
DRQ ← H
S107
BSY ← L
S108
IREQ ← L
S109
Check by host
END

Fig.5

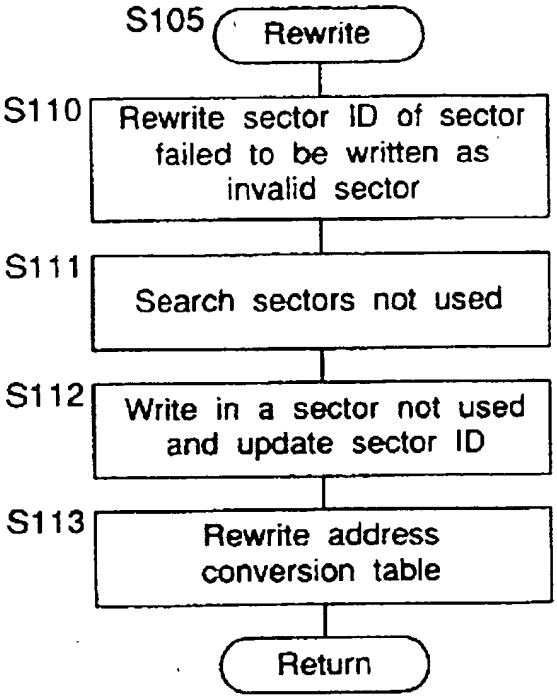

S105
Rewrite
S110
Rewrite sector ID of sector failed to be written as invalid sector
S111
Search sectors not used
S112
Write in a sector not used and update sector ID
S113
Rewrite address conversion table
Return

Fig.7

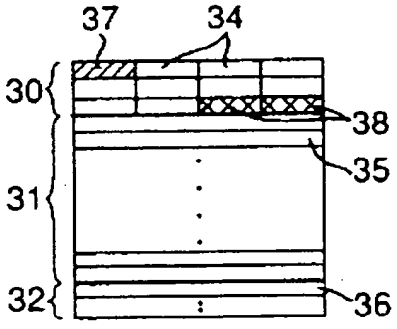

37
34
30
38
35
31
32
36

Fig.6

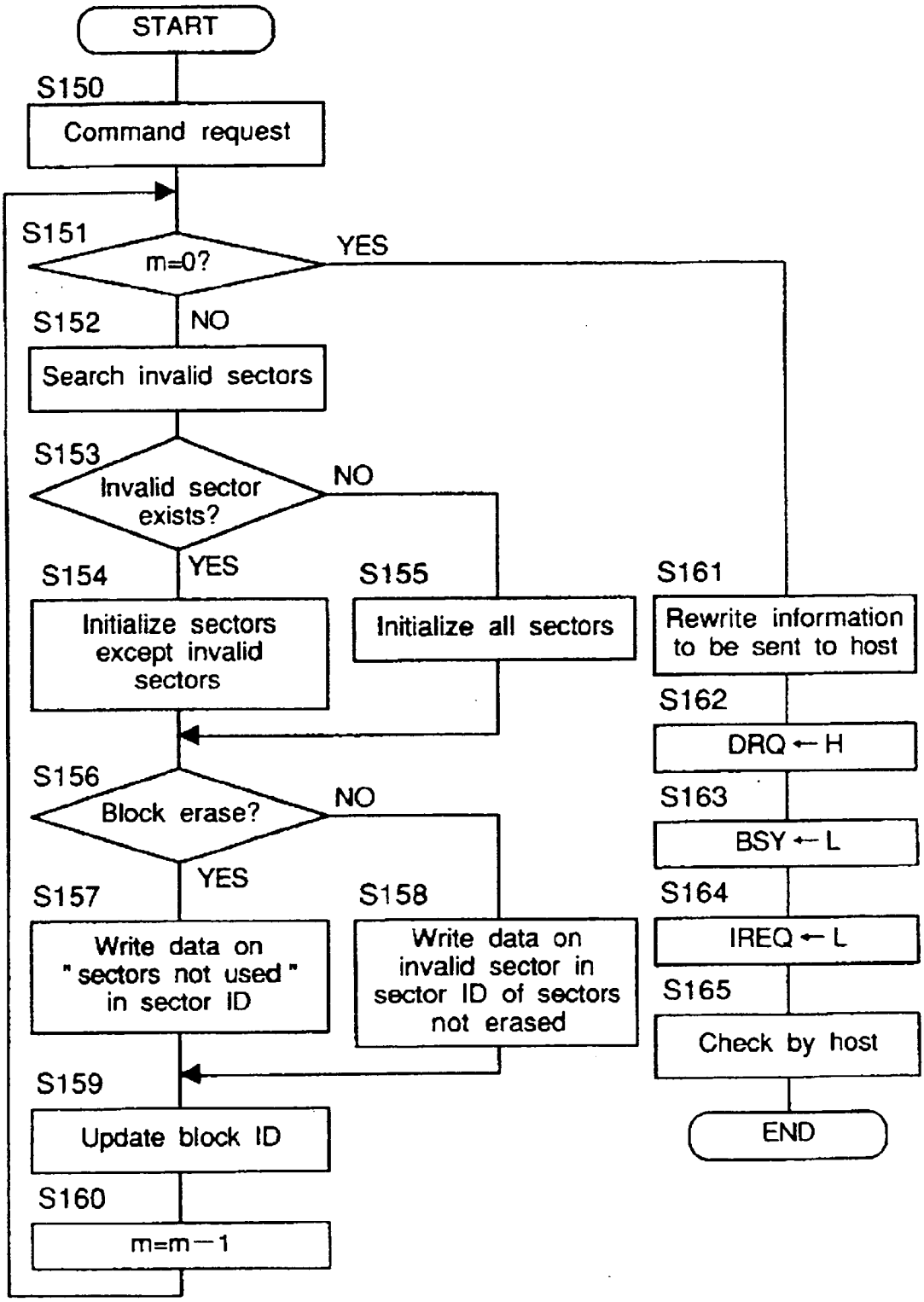

START
S150
Command request
S151
m=0?
YES
NO
S152
Search invalid sectors
S153
Invalid sector exists?
NO
YES
S154
Initialize sectors except invalid sectors
S155
Initialize all sectors
S156
Block erase?
NO
YES
S157
Write data on "sectors not used" in sector ID
S158
Write data on invalid sector in sector ID of sectors not erased
S159
Update block ID
S160
m=m−1
S161
Rewrite information to be sent to host
S162
DRQ ← H
S163
BSY ← L
S164
IREQ ← L
S165
Check by host
END

Fig.8

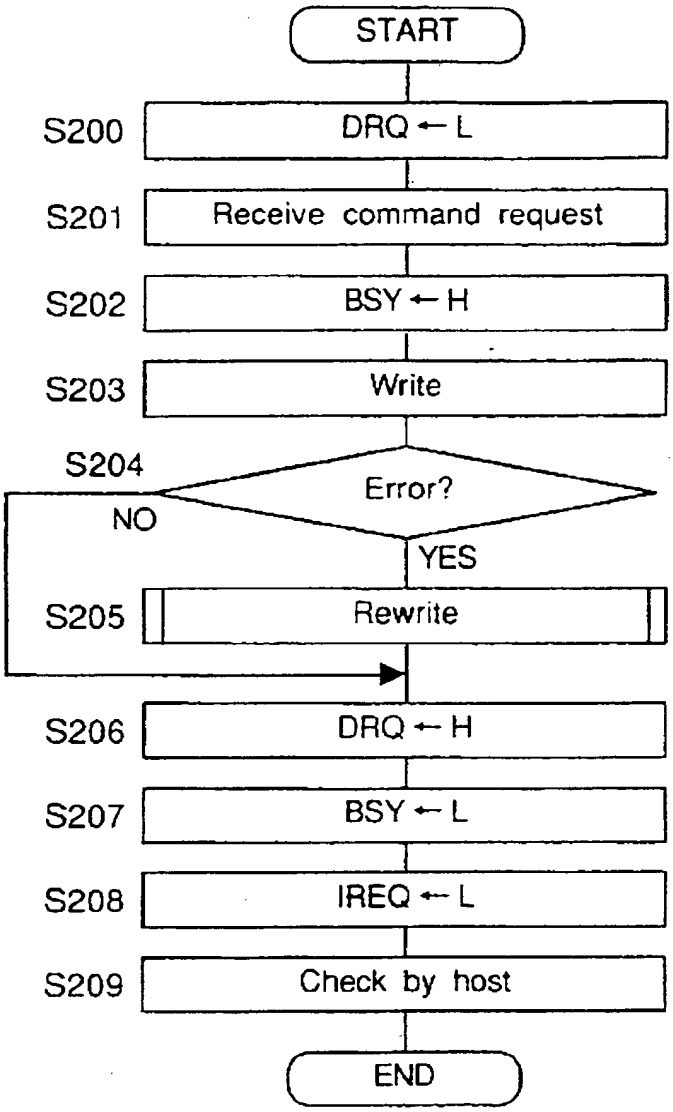

START
S200
DRQ ← L
S201
Receive command request
S202
BSY ← H
S203
Write
S204
Error?
NO
YES
S205
Rewrite
S206
DRQ ← H
S207
BSY ← L
S208
IREQ ← L
S209
Check by host
END

Fig.9

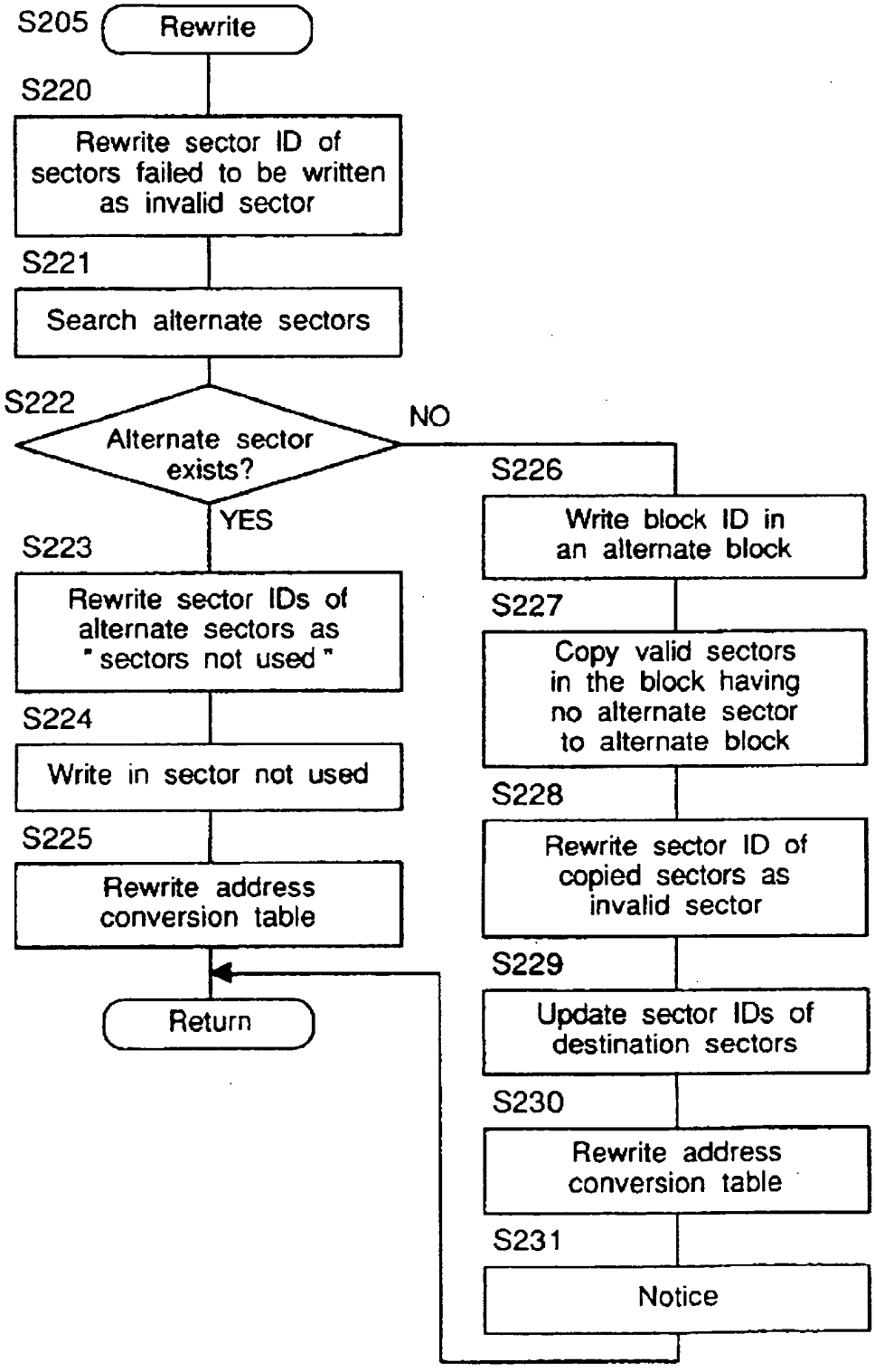

S205
Rewrite
S220
Rewrite sector ID of sectors failed to be written as invalid sector
S221
Search alternate sectors
S222
Alternate sector exists?
NO
YES
S223
Rewrite sector IDs of alternate sectors as " sectors not used "
S224
Write in sector not used
S225
Rewrite address conversion table
Return
S226
Write block ID in an alternate block
S227
Copy valid sectors in the block having no alternate sector to alternate block
S228
Rewrite sector ID of copied sectors as invalid sector
S229
Update sector IDs of destination sectors
S230
Rewrite address conversion table
S231
Notice

Fig.10

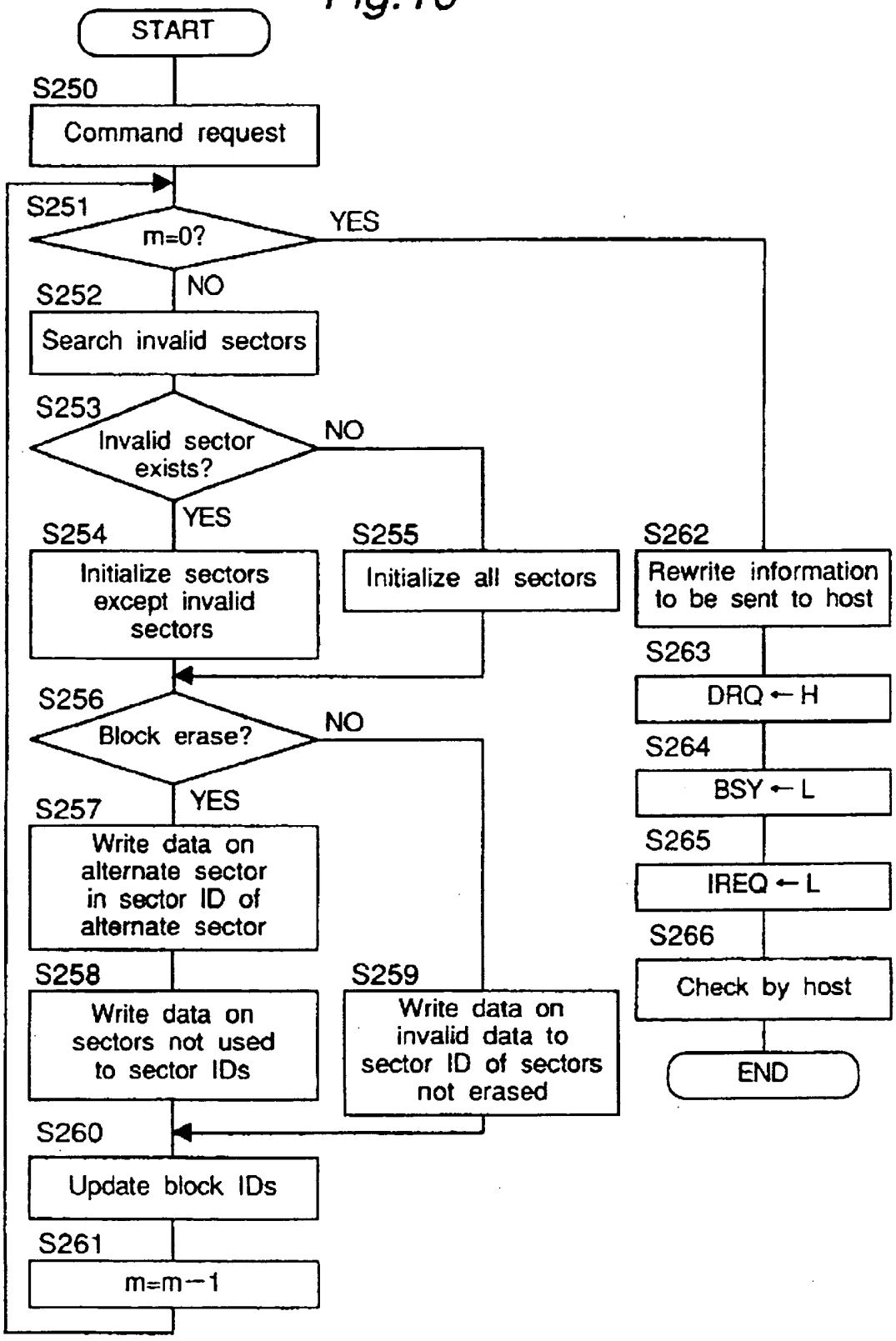

START
S250
Command request
S251
m=0?
YES
NO
S252
Search invalid sectors
S253
Invalid sector exists?
NO
YES
S254
Initialize sectors except invalid sectors
S255
Initialize all sectors
S256
Block erase?
NO
YES
S257
Write data on alternate sector in sector ID of alternate sector
S258
Write data on sectors not used to sector IDs
S259
Write data on invalid data to sector ID of sectors not erased
S260
Update block IDs
S261
m=m−1
S262
Rewrite information to be sent to host
S263
DRQ ← H
S264
BSY ← L
S265
IREQ ← L
S266
Check by host
END

Fig.11

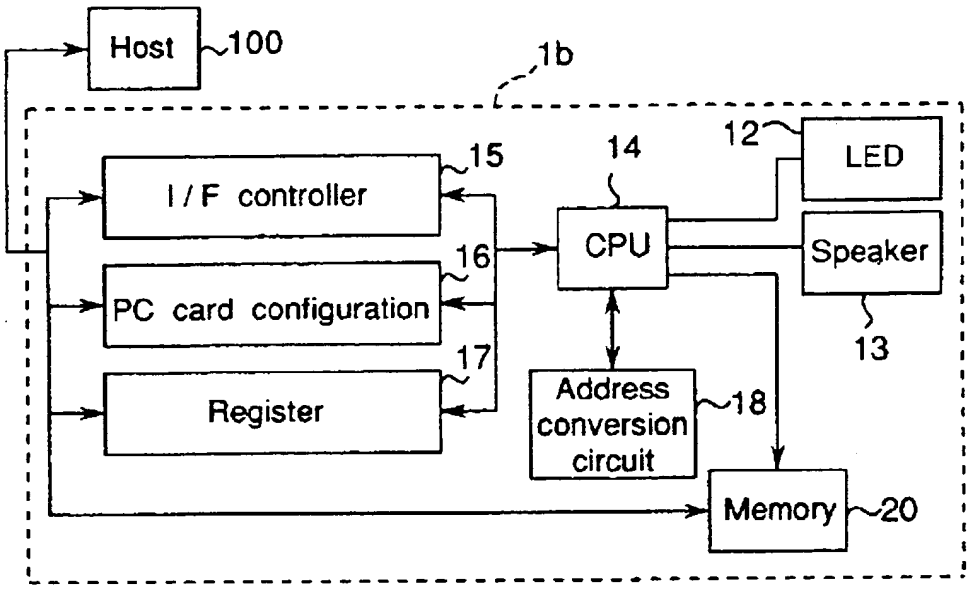

Host
100
1b
I / F controller
15
PC card configuration
16
Register
17
CPU
14
12
LED
Speaker
13
Address conversion circuit
18
Memory
20

Fig.12

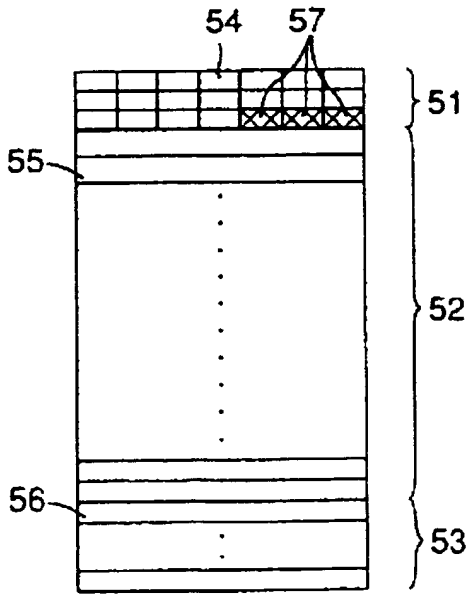

54
57
51
55
52
56
53

Fig.13

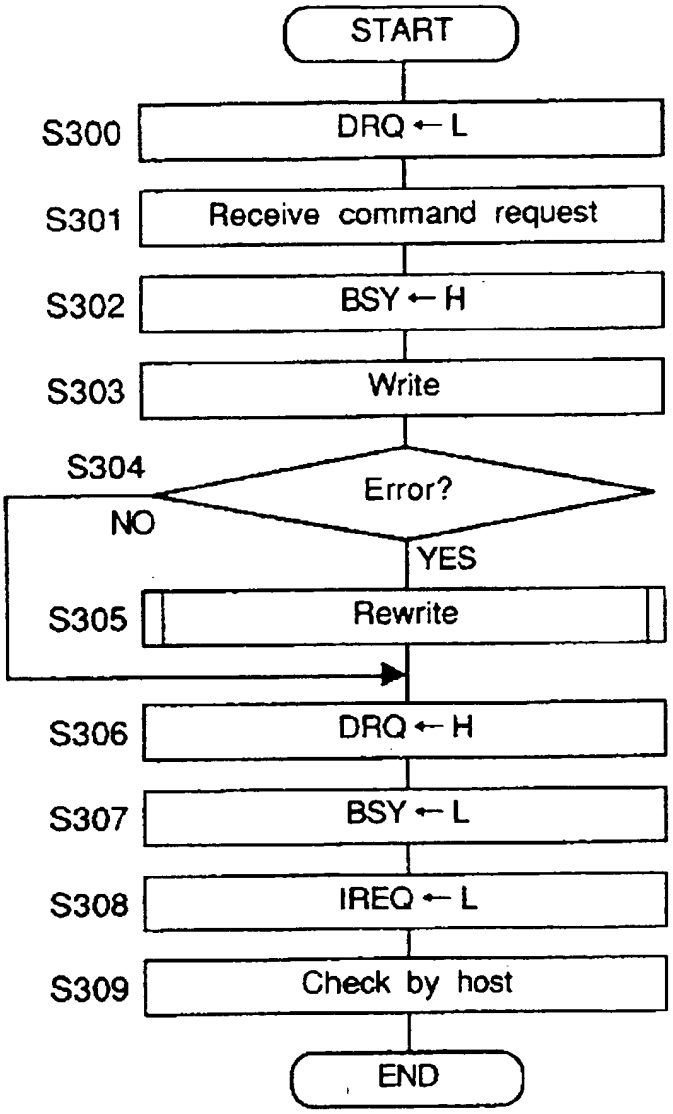

START
S300
DRQ ← L
S301
Receive command request
S302
BSY ← H
S303
Write
S304
Error?
NO
YES
S305
Rewrite
S306
DRQ ← H
S307
BSY ← L
S308
IREQ ← L
S309
Check by host
END

Fig. 14

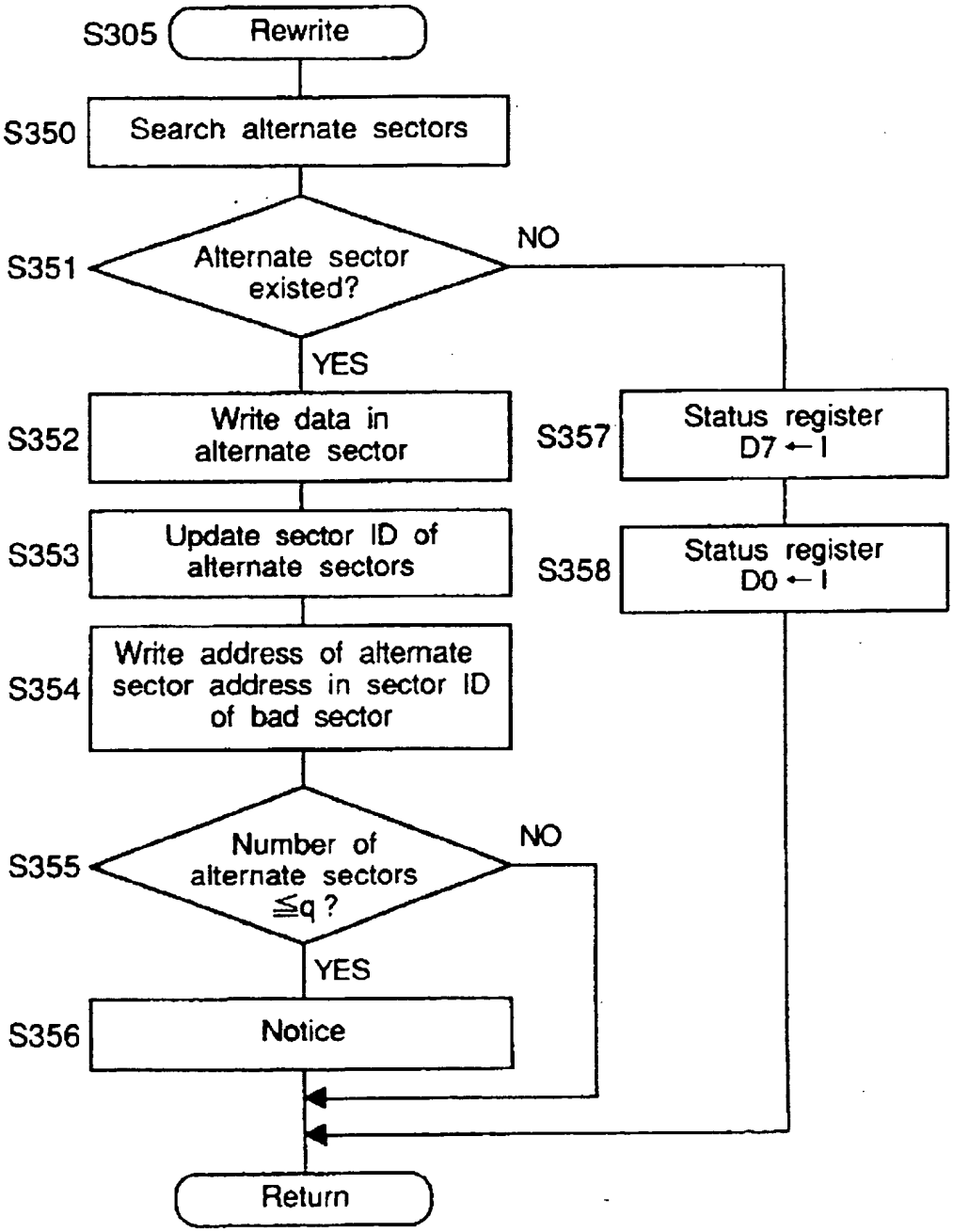

S305
Rewrite
S350
Search alternate sectors
S351
Alternate sector existed?
NO
YES
S352
Write data in alternate sector
S353
Update sector ID of alternate sectors
S354
Write address of alternate sector address in sector ID of bad sector
S355
Number of alternate sectors ≦q ?
NO
YES
S356
Notice
S357
Status register D7 ← 1
S358
Status register D0 ← 1
Return

Fig. 15

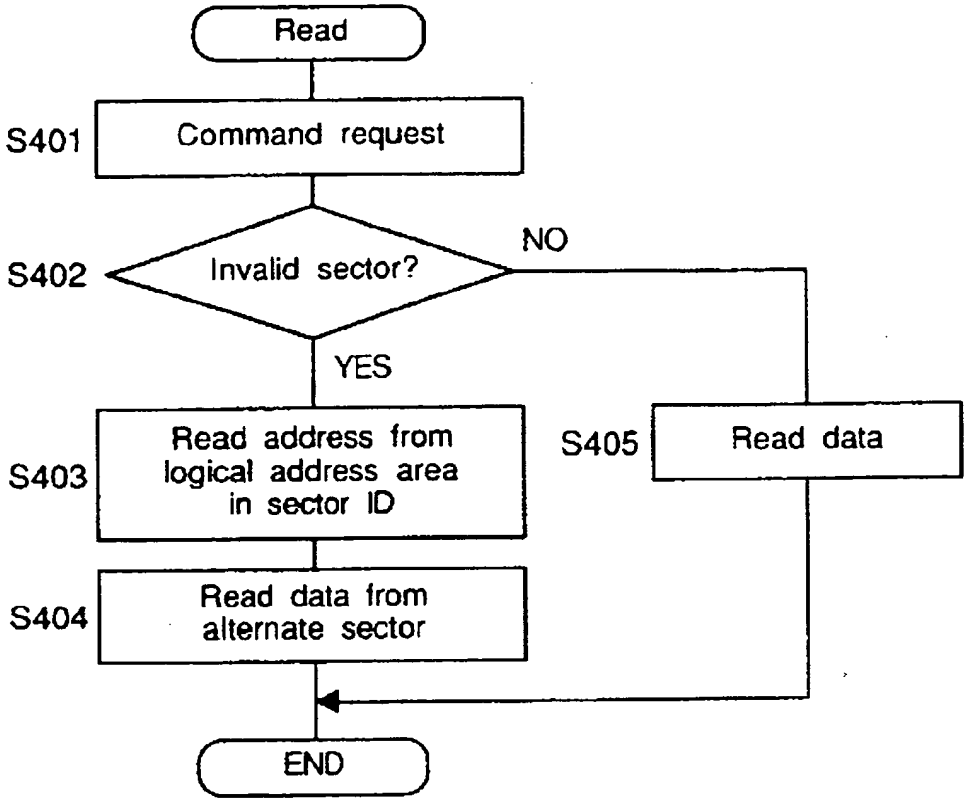

Read
S401
Command request
S402
Invalid sector?
NO
YES
S403
Read address from logical address area in sector ID
S404
Read data from alternate sector
S405
Read data
END

Fig. 16

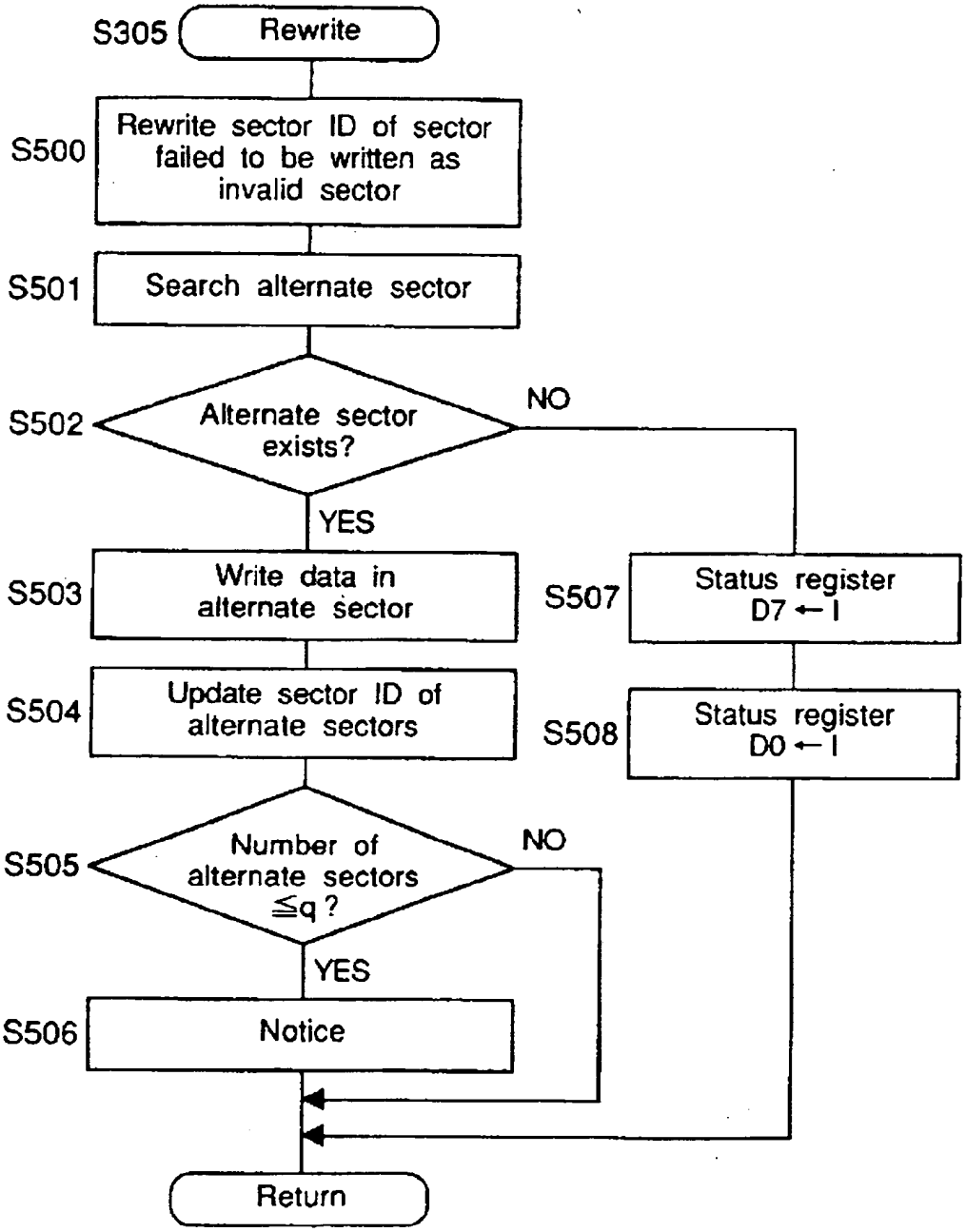

S305
Rewrite
S500
Rewrite sector ID of sector failed to be written as invalid sector
S501
Search alternate sector
S502
Alternate sector exists?
NO
YES
S503
Write data in alternate sector
S504
Update sector ID of alternate sectors
S505
Number of alternate sectors ≦q ?
NO
YES
S506
Notice
S507
Status register D7 ← 1
S508
Status register D0 ← 1
Return

Fig. 17

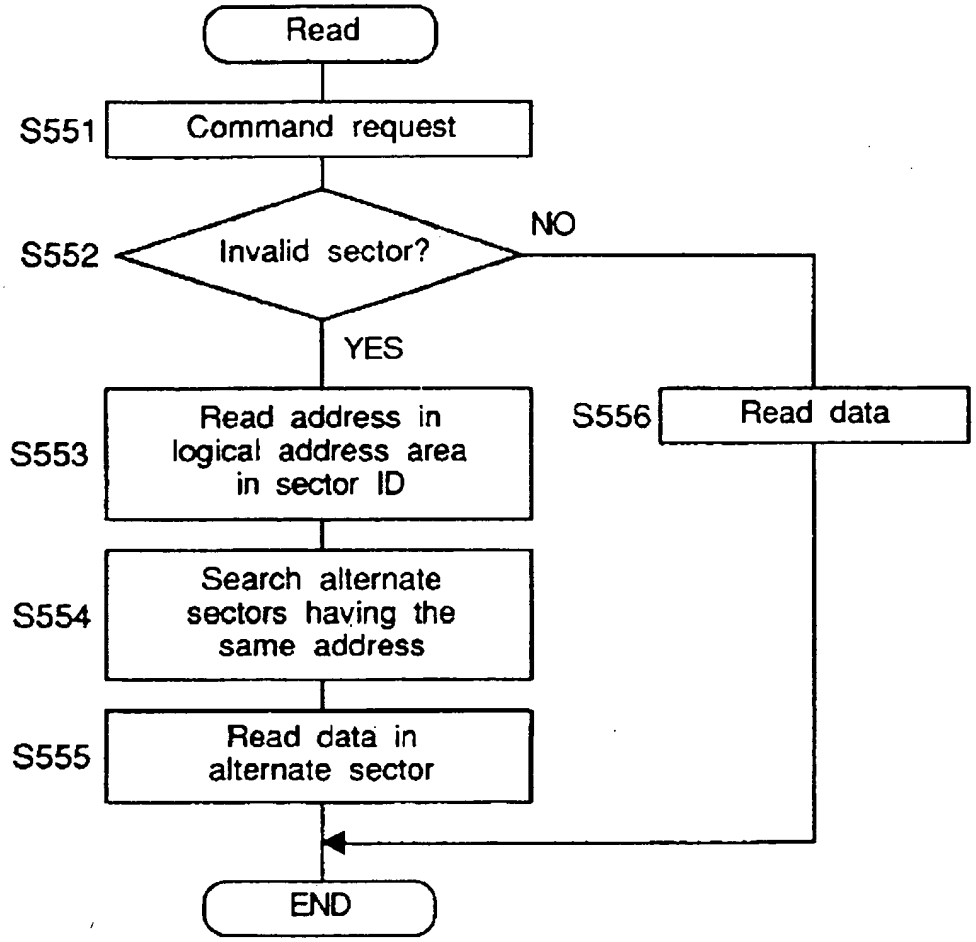
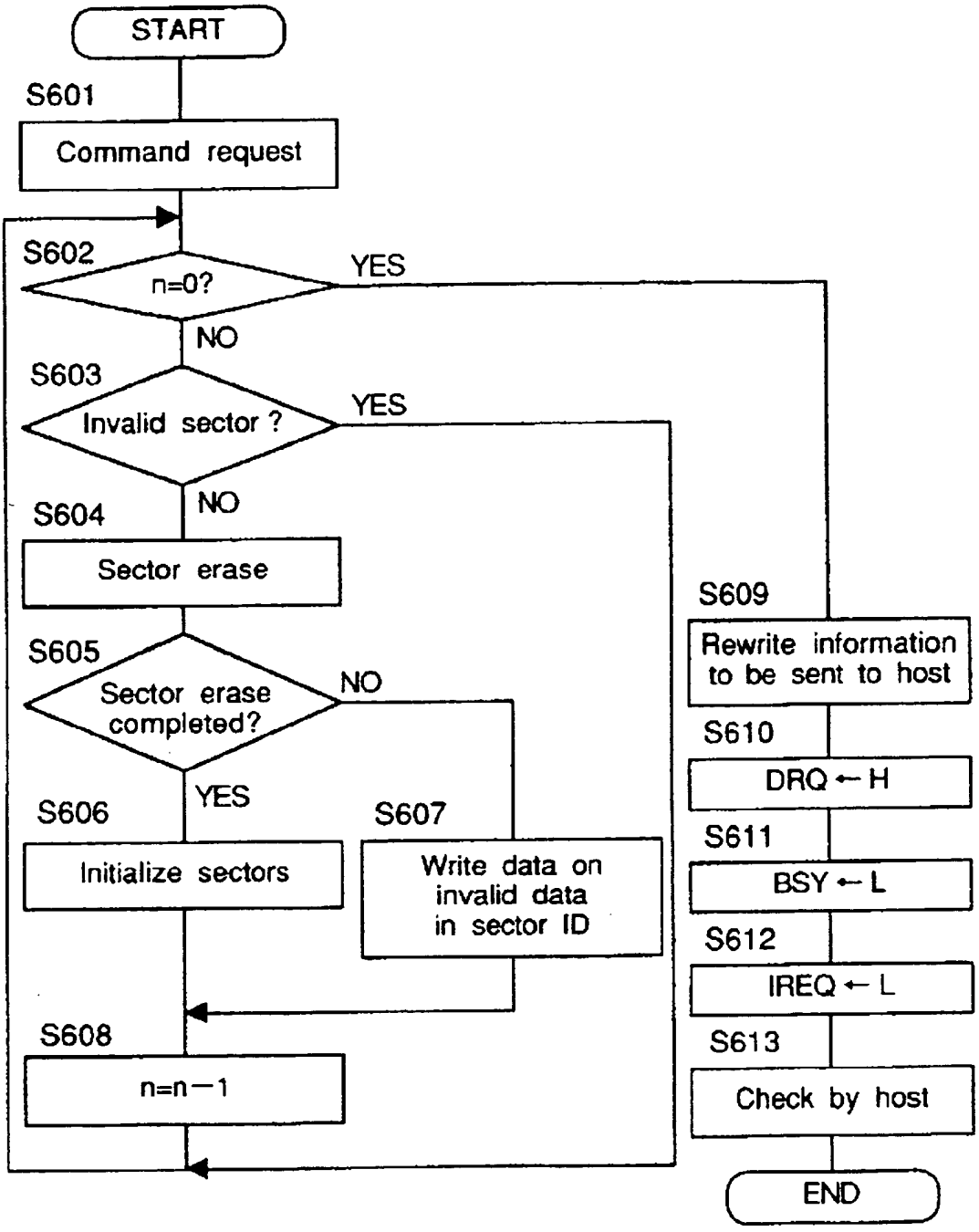

Read
S551
Command request
S552
Invalid sector?
NO
YES
S553
Read address in logical address area in sector ID
S554
Search alternate sectors having the same address
S555
Read data in alternate sector
S556
Read data
END

Fig. 18


START
S601
Command request
S602
n=0?
YES
NO
S603
Invalid sector ?
YES
NO
S604
Sector erase
S605
Sector erase completed?
NO
YES
S606
Initialize sectors
S607
Write data on invalid data in sector ID
S608
n=n−1
S609
Rewrite information to be sent to host
S610
DRQ ← H
S611
BSY ← L
S612
IREQ ← L
S613
Check by host
END

MEMORY DEVICE

The present invention relates to a memory device using a flash memory such as a memory card.

A flash memory is used for a semiconductor memory disk card (hereinafter referred to as PC card) used as a nonvolatile memory for reading and writing data. A PC card is connected to a host and operates as a memory medium therefor. By using a flash memory, the access speed of a PC card can be increased more than by use of a hard disk drive. A PC card using a flash memory is used according to PC Card Interface standard or the ATA (AT Attachment) standard in Japan Electronic Industry Development Association (JAIDA) which have been used as standards for a hard disk drive for a personal computer. By using the ATA standard, it is not necessary to limit a memory capacity or to use an exclusive driver provided for the PC card.

In the ATA standard, similarly to a hard disk drive, a flash memory has a memory space divided into a sector ID region and a sector region comprising sectors, and each sector in the sector region is managed by the sector ID region. A problem experienced when flash memory is used in a PC card is that

the number of rewrite operations that the flash memory can undergo is limited. Practically, write or rewrite of data from a host are repeated in particular sectors in a flash memory, and consequently, the life of the sectors becomes shorter than the other sectors. If a bad data write to a sector happens, the PC card warns the error to a host computer. The host which finds a bad data write decides that the PC card cannot be used. Even if the particular sectors are decided not to be used, the other sectors still can be used normally. However, the PC card cannot be used anymore.

An object of the invention is to provide a memory device used as a PC card which uses a memory therein efficiently..

To this end, the invention provides a memory device as set out in the claims.

In one aspect of the invention of a memory device, hen a data is written in a flash memory comprising a plurality of blocks, each block comprising a plurality of sectors, the block being a unit of erasing data, an address in a sector in the flash memory is identified according to address conversion in correspondence to a logical address, and when the data cannot be written in a first sector at the address identified, the data is written in a second sector in the same block as the first sector, without stopping due to the write error. Then, a physical address in the flash memory is rewritten in correspondence to the logical address in the address conversion table to an address of the second sector.

Then, a host accessing the memory device needs no more special processing for the write error. Similarly, when a data is written in a flash memory comprising a plurality of sectors, when the data cannot be written in a first sector at the address identified, the data is written in a second sector different from the first sector.

In a second aspect of the memory device of the invention comprising a flash memory, if each block in the flash memory comprises a plurality of sectors and a plurality of alternate sectors, when a data cannot be written in a first sector at the address identified, the data is written in an alternate sector in the same block as the first sector. Then, a physical address in the flash memory is rewritten in correspondence to the logical address in the address conversion table to an address of the second sector. Thus, a number of available sectors is kept constant. Further, a physical address in the flash memory is rewritten in correspondence to the logical address to an address of the second sector. Preferably, when all the alternate sectors have been used in the block the data and data which have been written to the block are written in a second block not used among the blocks. Similarly, when a data is written in a flash memory comprising a plurality of sectors, when the data cannot be written in a first sector at the address identified, the data is written in an alternate sector.

In third aspect of the invention of a memory device, the memory device comprises a flash memory comprising a plurality of sectors and a plurality of sector identification data (IDs) for storing status data on the sectors, and an address conversion means converts a logical address designated by a write command to a physical address in the flash memory. When the data cannot be written to a physical address in correspondence to the logical address, the data is written in a second sector, and a physical address in an address conversion table in correspondence to the logical address is rewritten to the physical address.

In fourth aspect of the invention of a memory device, the memory device comprises a flash memory, and the flash memory comprises a plurality of sectors, a plurality of alternate sectors and a plurality of sector identification data (IDs) for storing logical addresses for the sectors. An address conversion means converts a logical address designated by a write command to a physical address in the flash memory. When the data cannot be written to a physical address in correspondence to the logical address, the data is written in an alternate sector, and the logical address is stored in a sector ID in the sector IDs for the alternate sector.

An advantage of the present invention is that a life of the memory device can be extended.

Another advantage of the present invention is that a host connected to the memory device can access the memory device with no need of special commands.

These and other objects and features of the present invention will become clear from the following description taken in conjunction with the preferred embodiments thereof which are given by way of example and with reference to the accompanying drawings, and in which:

Fig. 1 is a diagram of a host and a PC card;

Fig. 2 is a diagram of a 64 kbytes type PC card;

Fig. 3 is a memory map of a memory;

Fig. 4 is a flowchart of write or overwrite when the memory is used in the PC card;

Fig. 5 is a flowchart of write processing;

Fig. 6 is a flowchart of initialization of the 64 kbytes type PC card using the memory;

Fig. 7 is a memory map of another memory;

Fig. 8 is a flowchart of write or overwrite when the memory is used in the PC card;

Fig. 9 is a flowchart of rewrite processing;

Fig. 10 is a flowchart of initialization of a 64 kbytes type PC card 1a using the memory;

Fig. 11 is a diagram of the 512 bytes type PC card;

Fig. 12 is a memory map of a memory;

Fig. 13 is flowchart of write or rewrite when the memory is used;

Fig. 14 is a flowchart of a first example of rewrite processing;

Fig. 15 is a flowchart of read processing;

Fig. 16 is a flowchart of a second example of overwrite processing;

Fig. 17 is a flowchart of read processing; and

Fig. 18 is a flowchart of initialization of the 512 bytes type PC card using the memory.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

There are two types of flash memories. A first type is a block type flash memory 10a which erases data in a size equal to or larger than a data transfer unit (512 bytes/sector) such as a size of 64 kbytes, while a second type flash memory 10b erases data in the unit of data transfer unit. Hereinbelow, a PC card using the block type flash memory is referred to as 64 kbytes type PC card 1a, while a PC card where data is erased in the data transfer unit is referred to as 512 bytes type PC card 1b.

Referring now to the drawings, wherein like reference characters designate like or corresponding parts throughout the several views, Fig. 1 shows a host computer 100 and a 64 kbytes type PC card 1a. The PC card 1a, 1b is inserted into a card slot in the host 100. The PC card 1a,

1b has an LED 2 and a speaker 3 for warning a write or rewrite error to a user. Embodiments of the 64 kbytes type PC card 1a and the 512 bytes type PC card 1b are explained below.

A. 64 kbytes type PC card

First, embodiments of the 64 kbytes type PC card 1a are explained. Fig. 2 shows a circuit of the 64 kbytes type PC card 1a. A command from the host 100 is received by a CPU 4 through an interface (I/F) controller 5, a PC card configuration section 6 and a register 7. A memory 10a or 10b included in the 64 kbytes card 1a is a flash memory which erases data in the unit of 64 kbytes (one block). If a write command to a logical address is received from the host 100, the CPU 4 identifies a physical address in correspondence to the logical address by referring to an address conversion table 8, and reads information stored in a sector ID (identification data) for the identified physical address. If the sector ID stores a data meaning that a data can be written to the physical address, the data from the host 100 is written to the sector. if a write error happens, a rewrite processing is performed to change a physical address to which the data is written. After data write or rewrite is completed, a sector ID of the physical address to which the data is written is updated.

The memory 10a or 10b is a flash memory which erases data in the unit of 64 kbytes (one block), and they have

different memory spaces between them. The memory 10a comprises a sector ID region and a sector region in a memory space thereof (Fig. 3), while the other memory 10b comprises an alternate sector region which will be used when bad data write happens, besides a sector ID region and a sector region in a memory space thereof (Fig. 7).

A-1. 64 kbytes type PC card using a first type of flash memory 10a

The PC card 1a with the flash memory 10a is explained. Fig. 3 shows a memory space for a block in the memory 10a. The memory space has two parts, a sector region 21 comprising a plurality of sectors 24 and a sector ID (identification data) region 20 storing sector IDs 23 each in correspondence to one of the sectors 24. Block IDs 22 are stored in a block ID region 22 (hatched in Fig. 3) in the sector ID region 20. The memory 10a erases data at a size in the unit of 64 kbytes (or one block) larger than the data transfer unit of 512 bytes. The block ID region 22 has a block information area wherein data on use status in a block are stored which show for example that a corresponding sector is not used or stores a data. Further, the sector IDs 23 stored in the region except the block IDs 22 have a logical address area, a sector information area and an error correcting code (CCD) area. The logical address area stores a logical address in correspondence to a physical address. The

sector information area stores data which means for example that a corresponding sector is used, is not used, or is an invalid sector which have caused a bad write.

Fig. 4 shows a flowchart performed by the CPU 4 when data is written or rewritten. First, D3 (data request signal DRQ) in a status register is reset at "L" level to advise to the host 100 that the card 1a is ready for a request of data write or rewrite from the host 100 (step S100). After receiving a write or rewrite command to a logical address from the host 100 (step S101), D7 (busy signal BUSY) in the status register is set at "H" level to process the command in the card (step S102). Then, a data is written or rewritten to a physical sector in the memory 10a in correspondence to a logical address received from the host (step S103). If a write error happens at the physical sector (YES at step S104), a rewrite processing explained later is performed (step S105). After the rewrite processing completes or a write error does not happen (NO at step S104), the status flag D3 (DRQ) in the status register in the 64 kbytes card 1a is set at "H" level meaning that the processing in the card completes (step S106), and the status signal D7 (BUSY) in the status register is set at "L" level (step S107). Then, an IREQ (interrupt request) signal is set at "L" level for permitting a check performed by the host 100 (step S109).

Next, the rewrite processing (step S105 in Fig. 4) is explained. In this processing, the CPU 4 writes the data in a different sector not used instead of the sector at which the data write failed. Thus, a life of the card 1a is elongated though the sector cannot be used. To the ID information sector for the sector, a data meaning an invalid sector is written in order to forbid access from the host 100. On the other hand, a data meaning that the sector is used to an ID information area of the different sector at which the data is written, and the address conversion table is changed so that the physical address in correspondence to the logical address received from the host 100 is the address of the different sector. Thus, even if a bad write happens, the host 100 can access the card 1a.

Fig. 5 shows a flowchart of the rewrite processing (step S105 in Fig. 4). First, a data meaning an invalid sector is written to a sector information area for the sector to which the data is failed to be written (step S110). Then, a sector for which a data meaning that the sector is not used is written in a sector information area therefor is searched in the sector IDs 23 (step S111). Then, a data is written to the searched sector not used, and a data meaning that the sector is used is written in a sector information area therefor in the sector IDs 23 (step S112). Then, the address conversion address 8 is rewritten so that a physical address

in correspondence to the logical address of the data is changed to the sector to which the data is written (step S113).

As explained above, when a write command at a logical address is received from the host 100, even if a bad write happens at a sector in the flash memory 10a in correspondence to the logical address, a data is written to a different sector not used. Thus, even if the data cannot be written to a part of the sectors in the flash memory 10a, it can be written without stopping operation in the PC card 1a due to happening of an error, and a life of the PC card 1a can be elongated.

Next, initialization of the 64 kbytes type PC card 1a is explained. The PC card 1a communicates with the host 100 by using an ATA command block for providing necessary control functions and status information. The command block is a general term representing seven registers, and it is provided in the register 7 (Fig. 2). An optional command is added to the commands supported in the ATA command block in order to initialize data in the memory 10a in this embodiment. When the memory 10a is used, the 64 kbytes type PC card 1a initializes the sectors or erases the data except the invalid sectors in accordance to the optional command. Then, a data meaning an invalid sector is written to a sector information area of the sector to which bad data erasing happens. After

the initialization, information on a memory capacity to be output to the host 100 is rewritten according to a number of the sectors which can be used. The host 100 performs write or rewrite based on the information, so that it can prevent access to invalid sectors.

Fig. 6 shows a flow performed by the CPU 4 when the above-mentioned optional command is received from the host 100. An initial value of a variable "m" used in the flowchart is a number of all the blocks in the memory 10a. After the optional command is received from the host 100 (step S150), the variable "m" designating a block to be processed is checked (step S151). If "m" is equals to zero (YES step at S151), or if a block to be processed is determined, it is searched if one or more sector IDs having a data meaning an invalid sector is written exist in the block (step S152). If it is decided that such a sector or sectors exist in the block (YES at step S153), data in the block is erased or initialized except the invalid sectors (step S154). If it is decided that such a sector does not exist in the block (NO at step S153), all the sectors in the block are erased or initialized (step S155). Next, if it is decided that the erasing completes normally (YES at step S156), a data meaning a sector not used is written in sector IDs for the sectors wherein the data are erased (step S157). On the other hand, if it is decided that bad data erasing happens (NO at step S156), a data meaning a

sector not used is written in sector IDs wherein the data are erased normally and a data meaning a invalid sector is written in sector IDs for the sectors wherein bad data erasing happens (step S158). The block IDs include an erase number area storing times of initialization of the block besides the block information area for writing information, for example, that the block is not used or written. Then, after completing the processing of step S157 or S158, the number stored in the erase number area is increased by one, and a data meaning that the block is not used is written in the block information area (step S159). Next, the variable "m" is decreased by one (step S160), and the flow returns to step S151 to repeat the processing of Steps S152 - S160 for a different block until erasing processing on all the blocks completes.

When the above-mentioned erasing processing on all the blocks completes or when the variable "m" becomes 0 (YES at step S151), a number of sectors not used is counted, and information such as cylinder number, head number and sector number to be sent to the host 100 in accordance to an identify drive command are rewritten (step S161). Then, a status flag D3 (data request signal DRQ) meaning completion of the processing in the card is set at "H" level (step S162), and a status flag D7 (busy signal BSY) is set at "L" level (step S163). Next, an interrupt request signal (IREQ) requesting

an interrupt to the host 100 is set at "L" level (step S164) for permitting a check by the host 100 (step S165).

As explained above, when a sector to which a data is written is changed due to an error, the address of a sector in correspondence to the logical address in the address conversion table is changed to the address of the sector to which the data is written. Further, relevant data in the sector area 20 are rewritten in correspondence to a change of sector. Therefore, the host 100 is not needed to have a special processing for such a PC card even if invalid sectors happen. For example, when the host 100 sends a write command, it sends no special command. When the host 100 accesses the PC card 1a, it sends an identify drive command to the PC card 1a. Then, the CPU 4 sends information on memory space available in the memory 10a such as cylinder number, head number and sector number to the host 100, and the host 100 gets information on the memory space which can be used.

A-2. 64 kbytes type PC card using a second type of flash memory 10b

Next, the PC card 1a with the flash memory 10b is explained. Fig. 7 shows a memory space in a block 33 in the memory 10b. The memory space thereof has three parts, and it is different from that shown in Fig. 3 in that the alternate sector region 32 is provided. A first part among the three

is a sector ID region 30. A block ID region 37 illustrated with hatching in the sector ID region 30 stores block IDs which have a block information area. In the block information area, status data status on the blocks are stored which means for example that the block is not used or is on use. Regions with crosses in the sector ID region 30 in Fig. 7 comprise alternate sector IDs 38 in correspondence to alternate sectors. Sector IDs are stored in the region 34 except the regions 37 and 38 illustrated with hatching and crosses in Fig. 7. Each of the sector IDs corresponds to a sector, and a sector ID 34 has a logical address area, a sector information area and an error correcting code (CCD) area. The logical address area stores a logical address in correspondence to a physical address. The sector information area stores data which show for example that a corresponding sector is on use, is not used, or is an invalid sector which causes a bad write. The second part is a sector region 31 comprising sectors 35 in correspondence to the sector IDs stored in the sector ID region 30. The third part comprises an alternate sector region 32 comprising a plurality of alternate sectors 36 to be used instead of sectors wherein bad write happens.

Fig. 8 shows a flow performed by the CPU 4 when data is written or rewritten. First, D3 (data request signal DRQ) in a status register in the 64 kbytes PC card 1a is reset at "L" level to advise to the host 100 that the card 1a is ready

for a request of data write or rewrite from the host 100 (step S200). After receiving a write or rewrite command to a logical address from the host 100 (step S201), D7 (busy signal BUSY) in the status register is set at "H" level to process the command in the PC card 1a (step S202). Then, a data is written or rewritten to a physical sector in the PC card in correspondence to a logical address received from the host 100 (step S203). If a write error happens at the physical sector (YES at step S204), a rewrite processing explained later (Fig. 9) is performed (step S205). After the rewrite processing (step S205) completes or a write error does not happen (NO at step S204), the status flag D3 (DRQ) in the status register in the PC card 1a is set at "H" level which means that the processing in the card completes (step S206), and the status signal D7 (BUSY) in the status register is set at "L" level (step S207). Then, an IREQ signal is set at "L" level for permitting a check by the host 100 (step S209).

As explained before, when the memory 10a is used in the PC card 1a, a data is written to a sector not used instead of a sector where a bad write happens. On the contrary, when the memory 10b is used in the PC card 1a in this embodiment, an alternate sector prepared beforehand in the alternate sector area 32 is changed to a state which can be used in the rewrite processing (step S205), and a data is written to the sector. Thus, a life of the PC card 1a can be made longer,

and a memory capacity to be used can be maintained constant. Further, when a bad write happens in a certain frequency, a write error is warned or notified to a user by emitting a light with the LED 2 or by generating a sound with the speaker 3.

Fig. 9 shows a flowchart of the rewrite processing (step S205 in Fig. 8). First, a data meaning an invalid sector is written in a sector information area for the sector to which the data is failed to be written (step S220). Then, an alternate sector is searched in the alternate IDs 38 in the sector ID area 30 wherein a data meaning alternate sector is written (step S221). If it is decided that such an alternate sector ID exists (YES at step S222), a data meaning a sector not used is written for initialization to a sector ID area in the alternate sector ID for the alternate sector (step S223). After the initialization of the alternate sector ID, a data is written to the alternate sector, and a data meaning a sector in use is written to a sector information area in a sector ID in correspondence to the sector to which the data is written (step S224). Next, the address conversion address 8 is rewritten, so that a physical address in correspondence to a logical address designated by the host 100 is changed to the sector to which the data is written at step S224 (step S225). Thus, the data is written to the alternate sector.

On the other hand, if it is decided that an alternate sector ID wherein a data meaning an alternate sector is stored does not exist in the alternate sector IDs 38 (NO at step S222), it is decided that all the alternate sectors in the block 33 is already used, and following processing is performed for writing the data to a different block not used. First, a data is written to a sector in a different block (hereinafter referred to as alternate block) having a block information area in the block ID 37 thereof to which a data meaning the block is not used is written, and a data meaning a sector is used is written to a sector information area in a sector ID 34 in correspondence to the sector to which the data is written (step S226). Then, data in the sectors in correspondence to sector IDs to which data meaning the sectors is used are written among the data in the block having no alternate sector are copied continuously in the alternate block (step S227). Next, a data meaning an invalid sector is written in sector information areas of the sector IDs 34 for the sectors copied in the alternate block at step S227 to prevent access (step S228). Then, the sector information area in sector IDs 34 for the alternate block is written with a data meaning that the sectors are used (step S229), and the content of the address conversion table 8 is changed (step S230). Finally, the LED 2 is lighted, and a speaker 3 is sounded to warn to a user that a bad write happens and the

capacity of the PC card decreases (step S231). The warning may be performed with only the LED 2 or the speaker 3.

Next, initialization of the memory 10b is explained. As explained above, when a bad write is notified to a user with the LED 2 and the speaker 3, the host 100 operated by the user sends an optional command for initializing the 64 kbytes type PC card 1a in correspondence to this notification. Then, the CPU 4 searches sector IDs to which a data meaning an invalid sector is written, initializes sectors except the sectors in correspondence to the sector IDs and rewrites information of memory capacity to be output in correspondence to an identify drive command received from the host 100.

Fig. 10 shows a flow performed by the CPU 4 when the above-mentioned optional command is received. An initial value of a variable "m" used in the flowchart is a number of all the blocks in the memory 10a. After the optional command is received from the host 100 (step S250), the variable "m" designating a block to be processed is checked (step S251). If the variable "m" is not zero (NO step at S251), or if a block to be processed is determined, it is searched if a sector ID or sector IDs having a data meaning an invalid sector is written exist in a block (step S252). If it is decided that such a sector ID or sector IDs exist in the block (YES at step S253), data in the block is erased or initialized except the invalid sectors (step S254). If it is decided that

such a sector ID does not exist in the block (NO at step S253), all the sectors in the block are erased or initialized (step S255). Next, if it is decided that the erasing or initialization completes normally (YES at step S256), a data meaning an alternate sector is written to sector information areas in alternate sector IDs for the alternate sectors to be erased normally (step S259), and a data meaning a sector not used is written to information areas in sector IDs (step S258). On the other hand, if it is decided that bad data erasing happens (NO at step S256), a data meaning an alternate sector is written to sector information areas of alternate sector IDs 38 for the sectors wherein the data are erased normally, a data meaning a sector not used is written to sector information areas of sector IDs 34, and a data meaning an invalid sector is written to sector IDs or alternate sector IDs for sectors wherein bad data erasing happens (step S259). The block ID 37 includes an erase number area storing times of initialization of the block besides the block information area for writing information, for example, that the block is not used or is on use. Then, after completing the processing of step S258 or S259, the number stored in the erase number area is increased by one, and a data meaning a block not used is written in the block information area (step S260). Next, the variable "m" is decreased by one (step S261), and the flow returns to step S251 to repeat the processing of Steps S252 -

S261 on a different block until the erasing processing on all the blocks completes.

When the erasing processing on all the blocks completes or when the variable "m" becomes 0 (YES at step S251), a number of sectors not used is counted, and information such as cylinder number, head number and sector number are rewritten (step S262). (When the host accesses the PC card 1a, it sends an identify drive command. Then, the CPU 4 sends information on memory space available in the memory 10a such as cylinder number, head number and sector number to the host 100, and the host 100 gets information on the memory space available.) Then, a status flag D3 (data request signal DRQ) meaning completion of the processing in the card 1a is set at "H" level (step S263), and a status flag D7 (busy signal BSY) is set at "L" level (step S264). Next, an interrupt request signal (IREQ) requesting an interrupt to the host 100 is set at "L" level (step S265) for permitting a check performed by the host 100 (step S266).

In the PC card explained above, alternate sectors are provided. When a write command at a logical address is received, even if a bad write happens at a sector in the flash memory 10b in correspondence to the logical address, a data is written to an alternate sector not used by setting an alternate sector ID for the alternate sector in a state where the alternate sector can be used. Thus, even if a data cannot

be written to a part of the sectors in the flash memory 10b, the data can be written without stopping operation due to the error, and a life of the flash memory can be elongated. It is to be noted that a number of sectors which can be used in the flash memory 10b can be kept constant.

Further, even when all the alternate sectors in a block is used, a data is written to a different block not used. Thus, a life of the flash memory can be elongated more.

When a sector to which a data is written is changed, the address of a sector in correspondence to the logical address in the address conversion table is changed to the address of the sector to which the data is written. Further, relevant data in the sector areas 30 and 32 are rewritten in correspondence to the change in sector. When a block is changed, relevant data in the sector areas 30 and 32 of the previous and alternate blocks are rewritten in correspondence to a change of block. Therefore, the host 100 is not needed to have a special processing for such a PC card 1a even if invalid sectors happen. For example, when the host 100 writes a data in the PC card, it needs no special command. When the host 100 accesses the PC card 1a, it sends an identify drive command to the PC card 1a to get necessary information on the memory space.

B. 512 bytes type PC card

Reference numbers in parentheses in Fig. 1 relates to a 512 bytes type PC card 1b which erases data in a data transfer unit (512 bytes/sector). The PC card 1b is inserted into a card slot in the host 100. The 512 bytes type PC card 1b has an LED 12 and a speaker 13 as means for notifying or warning a write or rewrite error to a user.

Fig. 11 shows a circuit of the 512 bytes type PC card 1b. A command from the host 100 is received by a CPU 14 through an I/F controller 15, a PC card configuration section 16 and a register 17. A memory 20 included in the 512 bytes card 1b is a flash memory which erases data in the unit of one sector (512 bytes). The CPU 14 reads information stored in a sector ID for a physical sector determined by an address conversion circuit 18 according to a logical address received from the host 100. If the sector ID stores a data meaning that a sector can be used, the data from the host 100 is written to the physical address. After the data is written, the sector ID for the sector is updated. If the physical sector to be written is an invalid sector which cannot be used, the CPU 14 performs one of rewrite processings explained later. If a number of the alternate sectors in the memory 20 is equal to or less than a threshold value "q", the LED 12 is lighted and the speaker 13 is sounded for notifying an error to a user.

Fig. 12 shows a memory space of the memory 20 which is a flash memory wherein erasing is performed in the unit of data transfer unit (512 bytes/sector). The memory space has three parts. A first part is a sector ID region 51 storing sector IDs 54. A sector ID 54 correspondence to a sector. The sector ID 54 has a logical address area, a sector information area and an error correcting code (CCD) area. The logical address area stores a logical address in correspondence to a physical address. The sector information area stores data which show for example that a corresponding sector is on use, is not used, or is an invalid sector which has caused a bad write. The second part is a sector region 52 comprising sectors 55. The third part comprises an alternate sector region 53 comprising alternate sectors 56 made of a plurality of alternate sectors 56 to be used instead of sectors wherein bad write happens. In the region 57 with hatching in the sector ID region 51 in Fig. 12 comprises alternate sector IDs in correspondence to the alternate sectors 56.

Fig. 13 shows a flow performed by the CPU 14 when data is written or rewritten. First, D3 (data request signal DRQ) in a status register in the PC card 1b is reset at "L" level to advise to the host 100 that the card 1b is ready for a request of data write or rewrite from the host 100 (step S300). After receiving a write or rewrite command to a

logical address from the host 100 (step S301), D7 (busy signal BUSY) in the status register is set at "H" level to process the command (step S302). Then, a data is written or rewritten to a physical sector in the memory 20in correspondence to a logical address received from the host 100 (step S303). If a write error happens at the physical sector (YES at step S304), a rewrite processing explained later is performed (step S305). After the rewrite processing completes (step S305) or a write error does not happen (NO at step S304), the status flag D3 (DRQ) in the status register is set at "H" level meaning that the processing in the card completes (step S306), and the status signal D7 (BUSY) in the status register is set at "L" level (step S307). Then, an IRQ signal is set at "L" level (step S308) for permitting a check by the host 100 (step S309).

In the first example of the rewrite processing (step S305 in Fig. 305), the CPU 14 writes a data in an alternate sector 56 not used instead of a sector wherein the data is not written normally, to maintain a number of sectors which can be accessed by the host 100 constant. Then, a life of the card 1b can be elongated. Further, if a number of available alternate sectors is equal to or less than a reference number "q", the LED 12 is lighted while the speaker 13 is sounded to alarm or notify it to a user. A data meaning an invalid sector is written to a sector information area of a sector ID

for the sector where a bad write happens, and an address of the alternate sector is written to a logical address area. Then, if a data meaning an invalid data is written to a sector information area of a sector ID for the sector to be accessed, the CPU 14 reads an address of the alternate sector written to a logical address area of the sector ID, and a data is written to or read from the alternate sector at the address.

Fig. 14 shows a flowchart of the above-mentioned rewrite processing (step S305 in Fig. 13). First, an alternate sector ID 57 is searched having a data meaning alternate sector (step S350). If it is decided that such an alternate sector ID exists (YES at step S351), a data meaning a sector not used is written to a sector ID area of the alternate sector ID 57 for initialization (step S352). After the initialization of the alternate sector ID, a data is written to a relevant alternate sector 56 (step S353). A data meaning an invalid sector is written to a sector information area of a sector ID for the sector wherein a bad write happens, and a physical address of the alternate sector 56 is written to a logical address area (step S354). On the other hand, if a sector in correspondence to a logical address required to be read by the host 100 is confirmed to be an invalid sector, it reads a data from an alternate sector at an address written in a logical address area of the sector ID therefor. Finally, if a number of available alternate sectors

56 is equal to or less than the reference number (YES at step S335), the LED 12 is lighted, and the speaker 13 is sounded to notify it to a user (step S356).

If it is decided that an alternate sector ID wherein a data meaning alternate sector is stored does not exist (NO at step S351), D7 (bad block register) in the error register is set at "1" (step S357), and D3 (error) in the error register is set at "1" (step S358).

Fig. 15 is a flowchart of read processing. First, a read command at a logical address is received (step S401). If a data meaning an invalid sector is written to a sector information area of a sector ID for a sector at a physical address determined by the address conversion circuit 18 according to the logical address (YES at step S402), an address written at a logical address area of the sector ID is read (step S403), and a data of an alternate sector at the address is read (step S404). If a data meaning an invalid sector is not written for the sector at the physical address determined by the address conversion circuit 18 (NO at step S402), a data at the sector is read (step S405).

B-2. A second example of rewrite processing

Next, a second example of the rewrite processing (step S305 in Fig. 13) is explained. In the second example, the CPU 14 writes a data meaning an invalid sector to a sector information area of a sector ID for the sector wherein bad

write happens. Then, a data is written to an alternate sector not used, and a logical address written to a logical address area of a sector ID for the sector wherein bad wrote happens is written to a logical address area of a sector ID for the alternate sector. Thus, a number of sectors which can be accessed by the host 100 is kept constant, and a life of the card can be elongated. Further, if a number of available alternate sectors is equal to or less than a reference number "q", the LED 12 is lighted while the speaker 13 is sounded to warn or notify it to a user. Then, if a sector accessed is an invalid sector, the CPU 14 searches an alternate sector ID 57 having a logical address area 56 wherein the same address as that for the invalid sector is written, and writes or reads a data to or from an alternate sector 56 in correspondence to the searched alternate ID.

Fig. 16 shows a flowchart of the above-mentioned rewrite processing (step S305 in Fig. 13). First, a data meaning an invalid sector is written to a sector information area of a sector ID for a sector where a bad write happens (step S500). Next, an alternate sector ID 57 is searched wherein a data meaning alternate sector is written (step S501). If it is decided that such an alternate sector ID exists (YES at step S502), a data is written to the alternate area 56 after data meaning that a sector is not used is written to a sector information area of the alternate sector

ID 57 for initialization (step S503). Then, a data meaning that the sector is used is written to the sector information area, and the same address as that of the logical address is written to the logical address area of the sector ID for the sector where bad write happens (step S504). Finally, if a number of available alternate sectors is equal to or less than a reference number (YES at step S505), the LED 12 is lighted, and the speaker 13 is sounded to notify or warn it to a user (step S506).

On the other hand, if it is decided that an alternate sector ID having a data meaning an alternate sector is stored does not exist (NO at step S502), D7 (bad block register) in the error register is set at "1" (step S507), and D3 (error) in the error register is set at "1" (step S508).

Fig. 17 is a flowchart of read processing. First, a read command at a logical address is received (step S551). If a data meaning an invalid sector is written to a sector information area of a sector ID for a sector at a physical address determined by the address conversion circuit 18 (YES at step S552), an address at a logical address area of the sector ID is read (step S553). Next, an alternate sector is searched having a logical address area wherein the same address as that read at step S553 is stored (step S554), and a data of the searched alternate sector is read (step S555). On the other hand, if a data meaning an invalid sector is not

written to the sector information area for the sector to be read (NO at step S552), a data at the sector is read (step S556).

Next, initialization of the memory 20 is explained. After receiving notification of occurrence of bad writes with the LED 12 and the speaker 13, if the host 100 operated by a user sends an optional command for initializing the 256 kbytes type PC card 1b in correspondence to this notification, the CPU 14 initializes sectors except the sectors in correspondence to the sector IDs having a sector information area to which data meaning an invalid sector is written and recognizes a memory capacity to be used practically. The information on the memory capacity is rewritten to be output in correspondence to an identify drive command received from the host 100. Then, the host 100 can obtain information on the memory space which can be used.

Fig. 18 shows a flow performed by the CPU 14 when the above-mentioned optional command is received. An initial value of a variable "n" in the flowchart is a number of the sectors in the card. After the optional command is received from the host 100 (step S601), the variable "n" representing a sector is checked (step S602). If the variable "n" = 0 does not hold (NO step at S602), of if n is determined, it is searched if a sector ID for the sector has a data meaning an invalid sector (step S603). If it is decided that the sector

is not an invalid sector (NO at step S603), data on the sector is erased (step S604). If it is decided that the erasing of the data in the sector completes normally (YES at step S605), the sector is initialized (step S606). That is, if the sector is an alternate sector, a data meaning alternate sector is written to a sector information area in the alternate sector ID for the alternate sector, while if the sector is a normal sector, a data meaning a sector not used is written to a sector information area of a sector ID for the sector (step S606). On the other hand, if it is decided that bad data erasing happens in the sector (NO at step S605), a data meaning an invalid sector is written to the sector ID or the alternate sector ID for the sector (step S607). Then, the variable "n" is decreased by one (step S261), and the flow returns to step S602 to repeat the processing of Steps S604 - S608 on a next sector until erasing processing on all the sectors completes.

When erasing processing on all the sectors completes (YES at step S602), a number of sectors not used is counted, and information such as cylinder number, head number and sector number is updated (step S609). Then, a status flag D3 (data request signal DRQ) meaning completion of the processing in the card 1b is set at "H" level (step S610), and a status flag D7 (busy signal BSY) is set at "L" level (step S611). Next, an interrupt request signal (IREQ) requesting an

interrupt to the host 100 is set at "L" level (step S612) for permitting a check performed by the host 100 (step S613).

The CPU 14 sends information on memory space available in the memory 20 such as cylinder number, head number and sector number to the host 100 in accordance to an identify drive command. As to the sectors where bad writes or bad erasing happen, a data meaning an invalid sector is written to sector information areas in sector IDs therefor. The host 100 gets information on the memory space available.

The rewrite processings on the 512 bytes type PC card 1b explained above are performed when a bad write happens. An address conversion table may be provided instead of the address conversion circuit 18, and a data may be written to a sector not used instead of the sector wherein a bad write happens. It is also possible to set one of the alternate sectors in a state which can be used in order that a memory capacity to be used is kept constant, while a data is written to a sector not used instead of the sector where bad write happens. In this case, the contents in the address conversion table is changed, or a physical sector in correspondence to a logical address designated by the host is changed to that of the sector to which the data is written. The initialization of the memory 20 is performed similarly to that shown in Fig. 18.

Although the present invention has been fully described in connection with the preferred embodiments thereof with reference to the accompanying drawings, it is to be noted that various changes and modifications are apparent to those skilled in the art. Such changes and modifications are to be understood as included within the scope of the present invention as defined by the appended claims unless they depart therefrom.

WHAT IS CLAIMED IS:

1. A memory device comprising:

a flash memory comprising a plurality of blocks, each block comprising a plurality of sectors, said flash memory erasing data in the unit of a block;

an address conversion table storing addresses of sectors in said flash memory in correspondence to logical addresses;

a data write means for identifying an address in a sector in said flash memory according to said address conversion table in correspondence to a logical address designated with a data write command including a data, and for writing the data in a first sector at the address identified or in a second sector different from the first sector, but in the same block as the first sector when the data cannot be written in the first sector; and

a rewrite means for rewriting a physical address in said flash memory in correspondence to the logical address in said address conversion table to an address of the second sector when said data write means writes the data in the second sector.

2. The memory device according to Claim 1, further comprising a data erase means for erasing data in said flash memory except sectors to which said data write means cannot write data, and for rewriting information on a memory space

of said flash memory which can be used, based on a number of sectors in which data are erased.

3. A memory device comprising:

a flash memory comprising a plurality of blocks, each block comprising a plurality of sectors and a plurality of alternate sectors, said flash memory erasing data in the unit of a block;

an address conversion table storing addresses of sectors in said flash memory in correspondence to logical addresses;

a data write means for identifying an address in a sector in said flash memory according to said address conversion table in correspondence to a logical address designated with a data write command including a data, and for writing the data in a first sector at the address identified or in an alternate sector among the alternate sectors in the same block as the first sector when the data cannot be written in the first sector; and

a rewrite means for rewriting a physical address in said flash memory in correspondence to the logical address in said address conversion table to an address of the alternate sector when said data write means writes the data in the alternate sector.

4. The memory device according to Claim 3, further comprising a data erase means for erasing data in said flash

memory except sectors to which said data write means cannot write data, and for rewriting information on a memory space of said flash memory which can be used, based on a number of sectors in which data are erased.

5. The memory device according to Claim 3, wherein when said data write means decides that all the alternate sectors have been used in a first block when the data is written to the first block, said data write means writes the data and data which have been written to the block in a second block not used among the blocks.

6. The memory device according to Claim 5, wherein said data write means comprises a data buffer, and when said data write means decides that all the alternate sectors have been used in a first block when the data is written to the first block, said data write means stores the data and data which have been to the block in said data buffer and stores the data continuously in said data buffer.

7. The memory device according to Claim 5, further comprising a data erase means for erasing data in the unit of block in said flash memory except sectors to which said data write means cannot write data, and for rewriting information on a memory space of said flash memory which can be used, based on a number of sectors in which data are erased.

8. The memory device according to Claim 5, further comprising a means for notifying to a user that said data write means write a data in a block not used.

9. A memory device comprising:

a flash memory comprising a plurality of sectors;

an address conversion table storing addresses of sectors in said flash memory in correspondence to logical addresses;

a data write means for identifying an address in a sector in said flash memory according to said address conversion table in correspondence to a logical address designated with a data write command including a data, and for writing the data in a first sector at the address identified or in a second sector different from the first sector when the data cannot be written in the first sector; and

a rewrite means for rewriting a physical address in said flash memory in correspondence to the logical address in said address conversion table to an address of the second sector when said data write means writes the data in the second sector.

10. The memory device according to Claim 9, further comprising a data erase means for erasing data in said flash memory except sectors to which said data write means cannot write data, and for rewriting information on a memory space

of said flash memory which can be used, based on a number of sectors in which data are erased.

11. A memory device comprising:

a flash memory comprising a plurality of sectors and alternate sectors;

an address conversion table storing addresses of sectors in said flash memory in correspondence to logical addresses;

a data write means for identifying an address in a sector in said flash memory according to said address conversion table in correspondence to a logical address designated with a data write command including a data, and for writing the data in a first sector at the address identified or for writing the data to an alternate sector in said flash memory when the data cannot be written in the first sector; and

a rewrite means for rewriting a physical address in said flash memory in correspondence to the logical address in said address conversion table to an address of the second sector when said data write means writes the data in the second sector.

12. The memory device according to Claim 11, further comprising a data erase means for erasing data in said flash memory except sectors to which said data write means cannot write data, and for rewriting information on a memory space

of said flash memory which can be used, based on a number of sectors in which data are erased.

13. A memory device comprising:

a flash memory comprising a memory space having a sector region comprising a plurality of sectors, a sector ID region comprising a plurality of sector IDs each storing status data and a logical address, and an alternate sector region comprising a plurality of alternate sectors;

an address conversion means for converting a logical address to a physical address in said flash memory;

a data write means for writing a data when a data of invalid sector is not stored in a sector ID for a sector at a physical address output by said address conversion means in correspondence to a logical address in a data write command, or for writing a data of invalid sector in the sector ID for the sector at the physical address when a data of invalid sector is stored in the sector ID and initializing an alternate sector not used, and writing the data in the alternate sector and writing a physical address of the sector in the sector ID therefor instead of a logical address stored in the sector ID; and

a data read means for reading data in a sector at the address stored instead of the logical address when a data of invalid sector is stored in a sector ID of a sector at an address in said flash memory output by said address conversion

circuit in correspondence to a logical address designated by the data read command.

14. The memory device according to Claim 13, further comprising a data erase means for erasing data in a sector having a sector ID therefor wherein data of invalid sector is not stored, writing data of sector not used in a sector ID of a sector to which data are erased normally, writing data of invalid sector to a sector ID for a sector to which data is not erased normally, and rewriting information on a memory space which can be used based on a number of sectors to which data of sector not used is written to sector IDs therefor.

15. The memory device according to Claim 13, further comprising a means for notifying to a user that the a number of alternate sectors not used in said flash memory is less than a predetermined value.

16. A memory device comprising:

a flash memory comprising a memory space having a sector region comprising a plurality of sectors, a sector ID region comprising a plurality of sector IDs each storing status data and a logical address, and an alternate sector region comprising a plurality of alternate sectors;

an address conversion means for converting a logical address to a physical address in said flash memory;

a data write means for writing a data when a data of invalid sector is not stored in a sector ID for a sector

at a physical address output by said address conversion means in correspondence to a logical address in a data write command, or for writing a data of invalid sector for the sector ID for the sector at the physical address when a data of invalid sector is stored in the sector ID and initializing an alternate sector not used, and writing the data to the alternate sector and writing a logical address stored in the sector ID for the sector to which data cannot be written to a sector ID for the alternate sector; and

data read means for reading a data in a sector having a sector ID storing data of the same logical address as that stored in a sector ID of the sector which cannot be used when data of invalid sector is stored in the sector ID for the sector at an address in said flash memory output by said address conversion means in correspondence to a data read command.

17. The memory device according to Claim 16, further comprising a data erase means for erasing data in a sector having a sector ID therefor wherein data of invalid sector is not stored, writing data of sector not used in a sector ID of a sector to which data are erased normally, writing data on invalid sector in a sector ID of a sector to which data is not erased normally, and rewriting information on a memory space which can be used based on a number of sectors to which data on sector not used is written in sector IDs thereof.

18. The memory device according to Claim 16, further comprising a means for notifying to a user that a number of alternate sectors not used in said flash memory is less than a predetermined value.

19. A memory device substantially as hereinbefore described with reference to any one of the accompanying drawings.

The Patent Office

43

Application No:	GB 9605873.0	**Examiner:**	Paul Nicholls
Claims searched:	1 - 19	**Date of search:**	29 May 1996

Patents Act 1977
Search Report under Section 17

Databases searched:

UK Patent Office collections, including GB, EP, WO & US patent specifications, in:

UK Cl (Ed.O): G4A (AEF)

Int Cl (Ed.6): G06F 3/06, 11/20

Other: Online: WPI

Documents considered to be relevant:

Category	Identity of document and relevant passage		Relevant to claims
X	EP 0,675,502 A2	(SUNDISK) - Whole document	1, 3, 9, 11, 13 and 16 at least
X	EP 0,618,535 A2	(SUNDISK) - Whole document	Ditto
X	EP 0,617,363 A2	(SUNDISK) - Whole document	Ditto
X	EP 0,597,706 A2	(SILICON STORAGE TECHNOLOGY) - Whole document	Ditto
X	EP 0,590,809 A2	(IBM) - Whole document	Ditto
X	EP 0,522,780 A2	(IBM) - Whole document	Ditto
X	EP 0,392,895 A2	(SUNDISK) - Whole document	Ditto

X	Document indicating lack of novelty or inventive step	A	Document indicating technological background and/or state of the art.
Y	Document indicating lack of inventive step if combined with one or more other documents of same category.	P	Document published on or after the declared priority date but before the filing date of this invention.
&	Member of the same patent family	E	Patent document published on or after, but with priority date earlier than, the filing date of this application.